Any Two Learning Algorithms Are (Almost) Exactly Identical

David H. Wolpert
NASA Ames Research Center
Moffett Field, CA 94035
dhw@ptolemy.arc.nasa.gov

October 4, 2000

Abstract

This paper shows that if one is provided with a loss function, it can be used in a natural way to specify a distance measure quantifying the similarity of any two supervised learning algorithms, even non-parametric algorithms. Intuitively, this measure gives the fraction of targets and training sets for which the expected performance of the two algorithms differs significantly. Bounds on the value of this distance are calculated for the case of binary outputs and 0-1 loss, indicating that any two learning algorithms are almost exactly identical for such scenarios. As an example, for *any* two algorithms A and B, even for small input spaces and training sets, for less than $2e^{-50}$ of all targets will the difference between A's and B's generalization performance exceed 1%. In particular, this is true if B is bagging applied to A, or boosting applied to A. These bounds can be viewed alternatively as telling us, for example, that the simple English phrase "I expect that algorithm A will generalize from the training set with an accuracy of at least 75% on the rest of the target" conveys 20,000 bytes of information concerning the target. The paper ends by discussing some of the subtleties of extending the distance measure to give a full (non-parametric) differential geometry of the manifold of learning algorithms.

1 Introduction

It has been a long-held goal of the learning community to limn the mathematical structure underlying supervised learning. This goal would be significantly advanced if we had a model-free geometry of learning, i.e., a geometry that depends only on the behavior of learning algorithms, not on how either they or target input-output distributions are parameterized. One crucial element of such a geometry would be a model-free metric, i.e., a non-parametric measure of the "distance" between (the behaviors of) any two learning algorithms. As demonstrated in this paper, given a measure of generalization performance (i.e., a loss function) there is a natural associated choice of such a metric.

As shown below, evaluating this metric for binary outpus and the zero-one loss function, one finds that the distance between any two learning algorithms is almost exactly zero. Given how the metric is defined, an immediate corollary is that only for an extremely small fraction of targets is the difference in the expected generalization performances of two learning algorithms more than vanishingly small. This emphasizes just how important prior information about the target is — unless you restrict the set of allowed targets to an extremely small set, any pair of algorithms you're considering will perform essentially identically. Stated differently, the amount of information concerning the target implicit in a simple statement like "My algorithm should perform substantially better than its opposite on the target" is quite large (i.e., the prior over targets must have very low entropy for the statement to be true).

To put this in context, one of the least important of the many ramifications of the no-free-lunch (NFL) theorems [10] is that averaged over all targets, the generalization performance of any two algorithms is exactly identical. The result of this paper adds depth to that ramification. It does so by establishing further that for almost every single target, the generalization error of any two algorithms A and B are almost exactly identical (for zero-one loss). So in particular, whereas the NFL theorems tell us (loosely speaking) that if B is boosting applied to A then B must perform worse than A as often as it beats A, the results of this paper indicate that it is for only a tiny fraction of the possible targets that the performance of A and B differ significantly at all.

Insofar as "intelligence" can be equated with ability to perform inductive inference, these results mean that, considered over all induction problems, there is very little difference between the intelligence of you and your dog. Not only must your dog outperform you 'as often' as vice versa —by the NFL theorems — but there are precious few problems in which your performances differ significantly.

Section 2 motivates and defines our metric. Section 2 presents our bounds on intelligence. Section 4 presents a preliminary foray into extending the metric to specify all other aspects of the geometry of learning, i.e., into providing a full differential geometry. Finally, Section 5 presents a discussion of some of the ways these results might be extended. It also elaborates how these resuls underscore what is perhaps the most glaring hole in the current theory of supervised learning.

2 The Distance Between Two Learning Algorithms

To have the discussion be of sufficiently broad scope we will employ the Extended Bayesian Formalism (EBF) [11, 7], introducing its salient aspects as we go. (The EBF both encompasses and reconciles Bayesian, sampling theory, and COLT analyses of learning, something not possible with any of the other mathematical framework that have been used to analyze supervised learning.) To begin, presume we have an input and output space, X and Y respectively, and let the random variables h_A and h_B be the two hypothesis input-output distributions generated by our two learning algorithms in response to a (random variable) training set d consisting of m pairs of $X - Y$ values. So the relevant behavior of A and B is fixed *in toto* by the two distributions $P(h_A \mid d)$ and $P(h_B \mid d)$. (In the EBF, since there are random variables for training sets and targets, exactly *how* a learning algorithm makes its guesses is irrelevant, as far as generalization performance is concerned.) Now there are many ways one might measure how different two learning algorithms A and B are. One, related to information geometry, is to use some information-theoretic functional of the two conditional distributions specifying the two algorithms. More formally, in terms of the EBF, we could (for example) note that $P(h_a, h_B) = \sum_d P(h_A, h_B \mid d)P(d) = \sum_d P(h_A \mid d)P(h_B \mid d)P(d)$. Using this we could measure the difference between A and B in terms of the mutual information of $P(h_A, h_B)$, i.e., the mutual information between the h_A and h_B made in response to d's generated in some canonical manner (e.g., by sampling all d's generated by all possible targets).

If we're given a loss function, another approach presents itself: find the two sets of priors over target input-output distributions f, $P(f)$, for which each of the algorithms is Bayes-optimal, and then measure the difference between those two sets. (E.g., if each set consisted of a single $P(f)$, one might measure the L^2 difference between those two $P(f)$'s, or a symmetrized Kullback-Liebler distance between them, or some such.) A slight variation is to find the $P(f)$'s that result in the best performance of the learning algorithms (rather than find the $P(f)$'s for which the algorithms are the best possible) and compare those.

If we're given a loss function though, then in some senses that function, and in particular its expectation value, provides a more important measure of how different two learning algorithms are than does some counterfactual optimizing $P(f)$, or even than do the hypotheses that the learning algorithms produce. This suggests that neither of the schemes sketched above is the best choice. One way to use a loss function more directly than those schemes would be to evaluate the mutual information between the loss of the two algorithms in response to all d's. However losses are real-valued numbers, so information-theoretic quantities like mutual information that don't reflect how close two loss values are, but only whether they're identical or not, are not necessarily the most meaningful measure of the relationship between the loss distributions of the two algorithms.

As an alternative, given a loss function, we can directly apply that function and evaluate the difference between (an appropriate function of) the expected generalization performances of the two algorithms. In other words, we can evaluate how differently the two algorithms perform on average for off-training-set (OTS) points [11, 10]. (Results that are almost identical to those presented in this paper hold for IID rather than OTS generalization error.) We can then directly use that difference in generalization performances to provide us with our distance measure. [1] The rest of this section presents a formal definition of this metric, the metric that will be investigated in the rest of this paper.

The EBF encompasses non-single-valued input-output hypotheses (i.e., such hypotheses taking the form of a full conditional distribution $P(y \mid x)$ rather than a single-valued function from X to Y) as well as non-single-valued targets. However here, for simplicity, restrict attention to single-valued hypotheses h_A and h_B. Also restrict attention to "vertical" likelihoods $P(d \mid f) = P(d_X, d_Y \mid f) = P(d_X)P(d_Y \mid d_X, f) = P(d_X)\prod_{i=1,m} P(d_Y(i) \mid d_X(i), f(d_X(i)))$, where d_X is the training set's (ordered) input values, and d_Y its (correspondingly ordered) output values [10]. As an example of such a likelihood, to simplify the notation, in this paper we'll consider single-valued noise-free f, so we can write f as a function from X to Y rather than a function from X to distributions over Y. This allows us to write $P(d_Y(i) \mid d_X(i), f(d_X(i))) = \delta(d_Y(i), f(d_X(i))$ for any $i \in \{1, ..., m\}$, where $\delta(A, B)$ is the generalized Kronecker delta function, which equals 1 if its arguments are equal, 0 otherwise. (This restriction on f doesn't change the results presented below; it just reduces the notational complexity.) We will also assume a uniform sampling distribution over X, so that $P(d_X)$ is uniform over all d_X. (While we do this primarily for convenience, it also seems quite natural for an investigation of the geometric structure relating learning algorithms.)

[1] This approach to defining the distance between learning algorithms is similar to the approach used to define an inner product between learning algorithms and posterior distributions over targets. See [11].

Let n be the number of elements in our input space (assumed finite for simplicity, as is the output space). As mentioned above, all that is relevant (as far as generalization is concerned) concerning learning algorithm A is the distribution $P(h_A \mid d)$. Accordingly, due to the finiteness of X and Y, any learning algorithm is a (finite-dimensional) vector living in a space of Cartesian products of unit simplices [12]. Label the manifold of that Cartesian product as S.

Define the generalization performance random variables $C_A(f,d) \equiv E(c \mid f,d,A)$ and $C_B(f,d) \equiv E(c \mid f,d,B)$. Here the function $c(f,h,d)$ is expected off-training set error for zero-one loss and our uniform sampling distribution: $c(f,h,d) \equiv \frac{\sum_{x \notin d_X} [1-\delta(h(x),f(x))]}{(n-m')}$, m' being the number of distinct pairs in d [12]. We require that the learning algorithms have no direct access to the target (although they may make *assumptions* concerning the target), so that $P(h \mid d,f) = P(h \mid d)$. Accordingly, for example, $C_A(f,d) = \sum_{h_A} P(h_A \mid d)\, c(f,h_A,d)$.

We want to have C_A and C_B provide us with our measure of the distance between A and B, i.e., provide us with our metric. Since metrics must be symmetric, we want our metric to be a function of $|C_A - C_B|$, $K(|C_A - C_B|)$, rather than (for example) $C_A - C_B$. Now for the purposes of this paper, we want our metric to reflect the full behavior of the algorithms, not just their behavior in response to one particular d. (Otherwise our metric would in effect be indexed by that d.) So we have to let d vary. Accordingly, we're led to consider C_A and C_B as depending on f and m, as in conventional sampling theory statistics, i.e., we're led to consider $E(K(|C_A - C_B|) \mid f,m)$. This is the expected difference between the generalization performances of the two algorithms for all training sets of size m sampled from the target f.

However just as for current purposes we don't want our metric to be indexed by d, we also don't want it to be indexed by f. Indeed, a natural quantity to consider concerning $E(K(|C_A - C_B|) \mid f,m)$ is the fraction of f for which it equals or exceeds some value ϵ, as a function of ϵ. A synopsis of such fractions, taking into account all possible ϵ, is the average over f of $E(K(|C_A - C_B|) \mid f,m)$, $D(m,A,B)$. It is this synopsis of the difference between the two algorithms A and B that provides us with our measure for how close A and B are.

More precisely, as with the conventional L^p norm of real analysis, to get a distance measure that obeys the triangle inequality (as all metrics must) we take $K(.)$ to be a positive monomial of the magnitude of its argument and take as our distance measure $D(m,A,B) \equiv K^{-1}[E(K(C_A - C_B) \mid m)]$. We can write this in full as

$$D(m, A, B) = K^{-1}[\ \sum_{f,d} P(d, f \mid m)\ K\{C_A(f, d) - C_B(f, d)\}\ \ ] \ . \qquad (1)$$

Here the average over f will be taken to be uniform, since this seems most natural in an investigation of the geometric structure of learning algorithms. (The extension to the case of non-uniform $P(f)$ is the subject of future work.)

To put this in perspective, recall that the NFL theorems tell us (among many other things) that $E(C_A - C_B \ \mid\ m) = 0$. So in essence, $D(m, A, B)$ is looking at higher moments of $C_A - C_B$. These moments must be non-zero in general; if they aren't, then for every f separately the two algorithms have exactly the same d-averaged generalization error as one another.

Extensions of the results of this paper to non-uniform sampling distributions, non-binary output spaces, different loss functions, non-single-valued hypotheses and/or targets (i.e., noise), regression rather than classification, etc. are all straight-forward. However they are beyond the scope of this paper.

3 Main Results

The simplest function $K(.)$ to consider is the squaring function, for which $D(m, A, B) = \sqrt{E((C_A - C_B)^2 \ \mid\ m)}$ where the average over f is uniform. An upper bound on this D can be calculated in closed form. Moreover, via Chebychev's inequality, that upper bound allows us to bound the fraction of all f such that the the difference in generalization error of two algorithms exceeds ϵ, as a function of ϵ.

To see all this, evaluate

$$\begin{aligned} E(K(C_A - C_B) \mid f, m) = & \\ & \sum_d P(d|f)\ [\sum_{h_A, h_B} P(h_A, h_B|d)\{c(f, h_A, d) - c(f, h_B, d)\}]^2 \\ \leq & \sum_{d, h_A, h_B} P(d|f) P(h_A, h_B|d)\ [c(f, h_A, d) - c(f, h_B, d)]^2 \\ = & \sum_{d, h_A, h_B} P(d|f) P(h_A|d) P(h_B|d)\ [c(f, h_A, d) - c(f, h_B, d)]^2 \ , \qquad (2) \end{aligned}$$

where the inequality follows from the convexity of the squaring operator.

To evaluate the associated value of D^2, pull its outer sum over f inside the W's sums over d, h_A, and h_B. Next plug in our likelihood and definition of c and split f into the union of the two sets $f(d_X)$ and $f(X \setminus d_X)$. In addition use the fact that $c(f, h, d)$ only depends on the values of f on $X \setminus d_X$, not on all of f. Doing all this reduces our bound to

$$\textstyle\sum_{d,h_A,h_B,f(d_X)} \; \delta(d_Y, f(d_X))\; P(d_X)\; P(h_A \mid d)\; P(h_B \mid d) \times \sum_{f(X \setminus d_X)} [c(f, h_A, d) \; - \; c(f, h_B, d)]^2 \; , \tag{3}$$

all divided by the number of target functions. In turn, for any d_X, we can write that number of target functions as the product $[\sum_{f(d_X)} 1]\;[\sum_{f(X \setminus d_X)} 1]$.

Consider any d, $f(d_X)$, h_A and h_B in the outer sum. Let $N(h_A, h_B, d)$ (or just N, for short) be the associated set of elements in $X \setminus d_X$ for which h_A and h_B disagree. (At the extreme, there are pairs of algorithms for which N is always all elements in $X \setminus d_X$, no matter what d is.) Now the values of $f(X \setminus d_X)$ on the elements in $X \setminus d_X \setminus N$ are irrelevant, since loss on those elements for h_A will equal the loss for h_B (whatever those loss values are), and will therefore cancel out when we subtract the two c's. So we can replace the summation operator $\sum_{f(X \setminus d_X)}$ with $\sum_{f(N)}$, if we multiply by $2^{|X \setminus d_X \setminus N|}$.

Writing $\frac{2^{X \setminus d_X \setminus N}}{\sum_{f(X \setminus d_X} 1} = \frac{1}{sum_{f(N)} 1}$ now leads us to consider the term $\frac{\sum_{f(N)} [c(f,h_A,d) - c(f,h_B,d)]^2}{\sum_{f(N)} 1}$. A moment's thought shows that this is equal to (the expectation of) the square of the difference between the number of heads and the number of tails in $|N|$ flips of an unbiased coin, all divided by $|X \setminus d_X|^2$ (since the two c's are each normalized to $|X \setminus d_X|$). Neglecting that division for the moment, the expected difference is the expectation of $[n_h - n_t]^2 = n_h^2 + n_t^2 - 2n_h(|N| - n_h)$, where n_h is the number of heads and n_t the number of tails out of the $|N|$ flips. When averaged this equals $4E(n_h^2) - |N|^2$ ($E(n_t^2) = E(n_h^2)$, and $E(n_h) = |N|/2$). The average of n_h^2 for $|N|$ samples of a Bernoulli process is just $|N|/4 \, + \, |N|^2/4$. So up to the overall divisor, our inner-most sum over $f(N)$ is just $|N|$.

Therefore $\sum_{f(X \setminus d_X)} [c_A(f, h_A, d) - c_B(f, h_B, d)]^2$, divided by $\sum_{f \setminus d_X} 1$, is $\frac{|N|}{|X \setminus d_X|^2} \leq 1/|X \setminus d_X|$. (Recall that the $2^{|X \setminus d_X \setminus N|}$ term gets cancelled by an identical term from the $\sum_{f(X \setminus d_X)} 1$ contribution to the overall normalization term $\sum_f 1$.) Note that this bound holds independent of h_A and h_B, and therefore independent of our learning algorithms, $P(h_A \mid d)$ and $P(h_B \mid d)$. As a result of this, our entire expression (including the normalization constant) is bounded above by

$$\sum_{d_X, f(d_X)} \frac{P(d_X)}{|X \setminus d_X| \; 2^{m'}} \; = \; \sum_{d_X} \frac{P(d_X)}{|X \setminus d_X|} \; < \; 1/(n - m) \; . \tag{4}$$

(By our uniform sampling distribution, $P(d_X)$ is independent of d_X.) Accordingly, since it is the square root of this quantity, D is bounded above by $1/\sqrt{n - m}$.

Notice that by the NFL theorems, the expected value of $C_A - C_B = 0$. Accordingly, D is the standard deviation of the distribution of values of $C_A - C_B$, the difference in generalization performance of our two algorithms. Moreover we can apply Chebychev's inequality to determine that the fraction of targets f for which $E(|C_A - C_B| \mid f, m) > \epsilon$ is bounded above by $(D/\epsilon)^2$. By our result for D, this bound is in turn bounded above by 1 / $[(n-m)\epsilon^2]$: the fraction of f for which $E(|C_A - C_B| \mid f, m) > \epsilon$ is bounded above by 1 / $[(n-m)\epsilon^2]$. Alternatively, we can use our result for D to bound the value a particular Bayesian quantity takes on under a uniform prior $P(f)$: the fraction of training sets d such that $E(|C_A - C_B| \mid d) > \epsilon)$ is bounded above by 1 / $[(n-m)\epsilon^2]$. [2]

Notice though that the probability that $n_h - n_t = z$ is simply the probability that in $|N|$ flips, $n_h = (|N| + z)/2$. In other words, it's Bernoulli distributed. This means we don't have to examine the distribution of the squares of z to perform closed-form calculations; we can examine the distribution of $|z|$ directly. Indeed, we can form a tighter bound on our fraction of targets by using $K(z) = |z|$. For this choice, W directly gives us the difference in generalization performance of A and B, so the fraction of all f for which $W(f, m, A, B)$ exceeds ϵ is just the fraction of f for which the (magnitude of the) difference in generalization performance of A and B exceeds ϵ. Employing the Hoeffding inequality to our coin-flip expression for this choice of W, we derive

$$P(f : E(|C_A - C_B| \mid f,\ m) \geq \epsilon)) \ < \ 2e^{-\epsilon^2 (n-m)/2} \ . \tag{5}$$

Again, we can instead use our bound to derive a Bayesian result: for uniform $P(f)$,

$$P(d : E(|C_A - C_B| \mid d) \geq \epsilon)) \ < \ 2e^{-\epsilon^2 (n-m)/2} \ . \tag{6}$$

As a simple example, take $n = 10^6$ (say 3 input dimensions, each of which can take on 100 values), and m = 1,000. This is a relatively small input space and training set. Nevertheless, in this situation, the fraction of targets on which the expected off-training-set 0-1 losses of any two algorithms differ from one another by more than one percent is less than $2e^{-49.95}$, which essentially equals $2e^{-50}$.

As another example, if in the same situation we expect that our favorite learning algorithm A has an expected generalization accuracy of at least 75%, that

[2]Given two learning algorithms A and B and some ϵ, in general there will both be priors $P(f)$ for which the quantity $P(d : E(|C_A - C_B| \mid d) > \epsilon)$ is larger than it is for a uniform prior, and $P(f)$ for which it is smaller. (N.b. for non-uniform $P(f)$ we can no longer equate the fraction of all d with the probability of some d, and must accordingly be careful to talk exclusively in terms of probabilities of d's rather than fractions of them.) For example, if A = {always guess the output $y \in Y = 0$, regardless of d and the query point $q \in X$} and B = {always guess $y = 1$ regardless of d and q} and if in addition $P(f)$ is the bi-modal distribution $(1/2)\delta(f, \{y = 1\ \forall x\})\ +\ (1/2)\delta(f, \{y = 0\ \forall x\})$, then the probability of a d such that $|C_A - C_B|$ = 1 is 1.

means that the algorithm B that always guesses the opposite of A has accuracy of at most 25%. Therefore the difference in accuracies exceeds 50%, and we have implicitly restricted f to a set consisting of $2e^{-125{,}000}$ of all f. That corresponds to approximately 20,000 bytes of information concerning our target. The little bit of English we used to describe the target sufficiently so that we believed that our algorithm would get at least 75% right conveys 20,000 bytes. This information reflects our prior knowledge concerning targets; the calculation presented here gives an inkling of just how much information is in that prior knowledge. Indeed, let us say that the learning algorithm produces trees, and let me say to you that "the target can be well-modeled by a shallow tree". Let's suppose that you can use that information to improve the algorithm's performance by another 5%. This results in a difference in performance (between your algorithm and the opposite-guessing algorithm) of 60%, and therefore increases the amount of information by a factor of 36/25. In other words, the simple phrase "the target can be well-modeled by a shallow tree" increases our information by about 8,000 bytes.

To provide a scale for these results, note that by the NFL theorems, $E(C \mid m)$ is independent of the learning algorithm for the uniform $P(f)$ considered here. Accordingly, to evaluate $E(C \mid m)$ we can pick the algorithm that always, for any d, picks a hypothesis h randomly according to a uniform distribution over the space of all possible hypotheses. Clearly for any particular f the associated OTS generalization error has to be 1/2, and therefore that must also be the f-averaged error. Accordingly, for the scenario considered in this paper, $E(C \mid m) = 1/2$, for any learning algorithm.

Now since the bounds derived in this paper apply to any two learning algorithms A and B, they apply in particular when B is this randomly-guess-h algorithm. In turn, that algorithm has expect error equal to that of algorithm A (by NFL). Accordingly, the quadratic $K(.)$ bound gives the standard deviation of the OTS generalization error of any learning algorithm A. Similarly, our bound for absolute value $K(.)$ tells us that for any learning algorithm A, the fraction of f for which the associated expected generalization error differs from its f-averaged value by more than ϵ is less than $2e^{-\epsilon^2(n-m)/2}$. So the examples above demonstrate that as soon as ϵ is significant on the scale of that f-averaged value (i.e., on the scale of 1/2), that fraction of f is tiny. For very exceedingly few f does expected error differ significantly from its f-averaged value.

To see what this means geometrically, view learning algorithms as vectors $C(f, d)$ with components indexed by (f, d) pairs. Then the NFL theorems state that all learning algorithms live on a simplex (for vertical likelihoods and the sort of loss functions considered in this paper). There are many aspects of the distribution of learning algorithms across that simplex and its dependence on $c(.,.,.)$ that one might be interested in (e.g., the symmetries of that distribution). In particular, the results of this paper tell us that for any two learning algorithm vectors on that simplex, $\vec{A}$ and $\vec{B}$, for only very few of the components i will the value of A_i differ substantially from the value of B_i.

4 Differential Geometry of Learning Algorithms

In general, our D need not be a full metric. Given that i) the triangle inequality is obeyed by our D; that ii) D is symmetric; and that iii) $D(m, A, A) = 0$ for any algorithm A, the remaining feature to check is whether iv) it is always true that $D(m, A, B) = 0$ implies that $A = B$. When (iv) holds we have a proper metric; when it does not, we instead have a distance measure of the sort encountered in relativity [8]. [3]

A simple example of when (iv) does not hold occurs when $m = 1$ and $n = 3$. For any particular d_X, have A guess uniformly randomly between two associated $X-Y$ functions. The first function has off-training set Y values (in some arbitrary canonical order) (1, 0). The second function has OTS Y values (0, 1). (Values of the two functions on the training set are irrelevant.) Have B also always guess randomly between two d_X-specified functions, but have those functions have as their OTS Y values the two sets (1, 1) and (0, 0). Then even though $A \neq B$, both C_A and $C_B = 1/2$, regardless of f or d, and therefore $D(1, A, B) = 0$.

Interestingly, although (iv) isn't always met, if our learning algorithms are both deterministic (i.e., always produce the same hypothesis in response to the same training set)) and f is single-valued, then we do indeed have a full metric. To see this first note that since the support of $P(f)$ is all f, for our likelihood $D(A, B) = 0$ implies that $C_A = C_B$ for every f and every d that lies on that f. So for every d_X and $f(X \setminus d_X)$, the h_A and h_B generated from training on $d = (d_X, f(d_X))$ must agree with $f(X \setminus d_X)$ just as often as each other. This is impossible unless $h_A(X \setminus d_X) = h_B(X \setminus d_X)$. Now without loss of generality, when there is no noise, we can restrict attention to learning algorithms that reproduce the training set exactly (there being no noise, any other algorithm is nonsensical). Accordingly, we see that the hypothesis input-output function produced by algorithm A in response to any training set d must be identical to the hypothesis function produced by algorithm B in response to that d, as claimed.

[3]In relativity the differential geometry is always locally governed by the Minkowski metric, which means in particular that any two points in space-time are separated by a distance of 0 even if they are distinct, so long as they are connected by a light-ray.

There is another way to ensure that (iv) is met, which is to change how we characterize algorithms. Formally, rather than as a full $P(h \mid d)$, we reduce our description of a learning algorithm to its associated $C(f, d) = \sum_h P(h \mid d)c(f, h, d)$. D can be written in terms of $C_A(f, d)$ and $C_B(f, d)$ rather than $P(h_A \mid d)$ and $P(h_B \mid d)$. Moreover, so long as $P(d, f \mid m)$ does not *exactly* equal 0 for any d and f, it follows immediately that $D(m, A, B) = 0$ implies that $C_A(f, d) = C_B(f, d)$. So if we work in the space of $C(f, d)$ (i.e., if we apply the non-invertible mapping replacing each $P(h \mid d)$ with its associated $C(f, d)$), we have a full and proper metric, meeting all conditions (i) through (iv).

The metric on a space does not fully specify the geometry of that space and the associated quantities like the curvature of the underlying space, the equation for geodesics across the space, etc. Somewhat confusingly though, the term 'metric' as commonly used can refer either to a distance measure like D, or instead to a differential geometry "metric" g, a quantity that *does* specify the geometry of the underlying space. Formally, such a g is a smoothly varying bilinear form[4] mapping any two vectors in a tangent vector space to the reals, one such form for the tangent space of each point on a manifold M. In contrast, a metric like D specifies a distance between any two points in some space. In general, such a quantity is even defined if the underlying space has no relation to a manifold [5, 8]. Accordingly, distinguish it from the other kind of metric, the differential geometry "metric" will henceforth be referred to as a 'bilinear field' (with it being implicit that that field varies smoothly over an underlying manifold).

In this paper, to help relate our metric to a bilinear field, we will express that metric as relating points on a manifold. So the first thing we must do is define that manifold which underlies both our metric and bilinear field. Since our metric relates learning algorithms, one natural choice for that manifold is the set S of possible $P(h \mid d)$. However to have property (iv) be met, for now we will instead define the manifold by viewing learning algorithms in terms of the associated C's. (This is similar to how we originally found it appropriate to define the metric in terms of differences of C's rather than in terms of information theoretic distances between $P(h \mid d)$'s.) Formally, the manifold M over which our metric will be defined is the set of possible (f, d)-indexed vectors $C(f, d)$.

Having both our measure D and our bilinear form g be defined voer the same manifold does not fully fix their relation. Formally, our D can be viewed as mapping pairs of points in M to the reals. However any bilinear field is instead a set of mappings, each taking pairs of vectors to reals, one such mapping for

[4]A bilinear form, sometimes called a non-degenerate sesquilinear product, is an inner product just without the requirement that the image of the product be non-negative.

the tangent space of every point on M. These two quantities concern different mappings, operating over different spaces.

Conventionally in differential geometry, the metric and the bilinear field are related in the definition of infinitesimal path length. The basic idea starts with the fact that, by definition of manifold, surrounding any point $u \in M$ there must be a neighborhood $N(u)$ of points in M that is continuously bijectively mapped (via a coordinate system $\psi(.)$) into $E(u)$, a (perhaps infinitesimal) subregion of a Euclidean space. The tangent space at u, as expressed in the coordinate system $\psi(.)$, is the set of all vectors that are tangent at $\psi(u)$ to (the $E(u)$-image of) a path in M that passes through u. That tangent space is isomorphic to $E(u)$, and we can use that space to relate any two points u' and u'' such that $\psi(u), \psi(u')$, and $\psi(u'')$ are infinitesimally close (and therefore both in $N(u)$. We do this by writing in the usual way $\psi(u') = \psi(u'') + v$ for a tangent vector v.

This allows us to define the distance from u' to u'' in terms of the norm of v, which can in turn be defined using inner products. Formally, we take the square of the path length for the infinitesimal path from $\psi(u')$ to $\psi(u'')$ — the square of the value given by the metric for the distance between those two points on M — to be the square of the inner product of the tangent vector v with itself, where that inner product is given by $g(u)$, the bilinear field evaluated at u. Note that we cannot do this for arbitrary metrics. The metric must be of a particular form for any points u' and u'' infinitesimally close to one another — it must be a norm-induced metric, with the norm in turn being induced by an inner product. Note also that it is only because u' and u'' are infinitesimally close to u and because g is assumed smoothly varying that we can uniquely write an associated $g(u)$. This approach cannot be used for non-infinitesimal v. This means in particular that while this approach can be used to go from a metric to a bilinear field, it cannot be used to uniquely go from a bilinear field to a metric over the entire manifold. In general, one must specify a path of integration connecting any two points on the manifold to evaluate a distance between those points by means of a bilinear field.

Writing it out, with $R(.,.)$ the square of the metric giving distances between pairs of points on M, we must be able to write $R(u', u'') = \sum_{i,j} [g(u)]_{ij} v^i v^j$, where the matrix $g(u)_{ij}$ is our putative bilinear field evaluated at u and expressed in the coordinate system ψ, and the upper indices on v delineate its component values in the coordinate system ψ: $v^i \equiv [\psi(u')]^i - [\psi(u'')]^i$. So long as $g(u)_{ij}$ is non-degenerate (i.e., has non-zero determinant) and smoothly varying with u, it meets all the requirements of a bilinear field, so that we have indeed gone from a metric to a bilinear form. To evaluate $g(u)$ in the ψ coordinate system, choose

$v^i = \alpha\delta^i_p + \beta\delta^i_q$. Then $\sum_{i,j}[g(u)]_{ij}v^iv^j = \alpha^2[g(u)]_{pp} + \beta^2[g(v)]_{qq} + 2\alpha\beta[g(u)]_{pq}$. So as long as both ψ^{-1} and $R(.,.)$ are doubly differentiable at u, we can write $[g(u)]_{pq} = \frac{1}{2}\frac{\partial^2}{\partial\alpha\partial\beta}R[\psi^{-1}(v + \psi(u)), u]|_{\alpha=\beta=0}$.

For us, $D(m, A, B) = K^{-1}\{\sum_{f,d} P(d, f \mid m)K[C_A(f, d) - C_B(f, d)]$. For quadratic $K(.)$, this is explicitly in the general form of a loss-induced metric, with u' and u'' given (in our coordinate system) by two infinitesimally close $C(f, d)$'s, and g_{ij} given by $P(d, f \mid m)$. Furthermore, so long as there is no f and d for which $P(f, d)$ equals zero *exactly*, that $g(u)_{ij}$ is non-degenerate. Note that this $g(u)_{ij}$ is explicitly independent of position on the manifold, i.e., it does not vary with changes to $C_B(f, d)$. Therefore, trivially, it is a smoothly varying function of position on the manifold, and so meets all the formal requirements for a bilinear field.

Because our bilinear field is independent of position on the manifold, our space is flat (the Christoffel symbols all vanish). In addition, that bilinear field is diagonal in our coordinate system, by inspection (we don't have a double sum over (f, d) pairs). Finally, for the uniform $P(f)$ and $P(d_X)$ considered here, $P(f, d)$ is uniform. So the underlying manifold is actually Euclidean, for our $P(f, d)$. For other noise processes, sampling distributions, and/or priors over targets, in general the manifold will be non-Euclidean. Of course, in general all of this would vary if we changed $K(.)$ or in some other way altered our metric. In particular, such alternations could result in non-flat geometries, in which geodesics are curved.

As an alternative to the foregoing, we could consider the case where the underlying manifold M is the set of $(h - d)$ indexed vectors $P(h \mid d)$. For this choice of underlying manifold property (iv) of a metric is violated for our D. On the other hand, now the vector characterizing a particular learning algorithm will not vary if we change the choice of loss function $c(., ., .)$, i.e., learning algorithms and loss functions aren't conflated. For this choice of underlying manifold and for quadratic $K(.)$ we can directly expand

$$K[D(m, A, B)] = \sum_{(h,d),(h',d')} g_{(h,d),(h',d')}\;[A_{(h,d)} - B_{(h,d)}]\;[A_{(h',d')} - B_{(h',d')}]\;, \qquad (7)$$

where $A_{h'',d} \equiv P(h_A = h'' \mid d)$ and similarly for $B_{h'',d}$, and where g can be written in explicitly symmetric form as

$$g_{(h,d),(h',d')} \;\equiv\; \sum_f c(f, h, d)\; c(f, h', d')\; \delta(d', d)\; \sqrt{P(d, f \mid m)P(d', f \mid m)}\;. \qquad (8)$$

Whether this g is non-degenerate will depend on $P(d, f \mid m)$ and $c(., ., .)$, in general. For the choices of those quantities considered in this paper, if non-deterministic learning algorithms are allowed, then this g can be degenerate. To see this, first note that for any four learning algorithms given by $P(h_A \mid d)$, $P(h_B \mid d)$, $P(h_F \mid d)$, and $P(h_G \mid d)$, we can write

$$\sum_{(h,d),(h',d')} g_{(h,d),(h',d')}[A_{h,d} - B_{h,d}][F_{h',d'} - G_{h',d'}]$$
$$= \sum_{f,d} P(d, f \mid m)[C_A(f,d) - C_B(f,d)][C_F(f,d) - C_G(f,d)]\ . \quad (9)$$

So choose $P(h_A \mid d)$ and $P(h_B \mid d) \neq P(h_A \mid d)$ so that $C_A(f,d) = C_B(f,d)$. (Recall the example of of such a case presented above.) Then $C_A(f,d) - C_B(f,d) = 0$ and therefore the inner product between $[C_A(f,d) - C_B(f,d)]$ and any other difference between a pair $C(f,d)$'s must equal zero. Accordingly, the inner product between $[A_{h,d} - B_{h,d}]$ and some other tangent vector equals zero, even though $[A_{h,d} - B_{h,d}] \neq 0$. Therefore $g_{(h,d),(h',d')}$ is degenerate.

Regardless of the value of its determinant, due to its manifestly non-diagonal nature, this new $P(h \mid d)$-based g is not Euclidean in the coordinate system we've used to define it. However this new g is necessarily flat (it does not vary as one moves over the underlying manifold), just like the g for the case where M consists of vectors $C(f,d)$. In general, regardless of our choice of underlying manifold, to have that manifold not be flat the distance between two learning algorithms $\vec{A}$ and $\vec{B}$ cannot be uniquely fixed by the difference $\vec{F} \equiv \vec{A} - \vec{B}$. For example, if the distance varies with changes to $\vec{A}$ that leave $\vec{F}$ unchanged, then in general the manifold is not flat. Generically, such non-flatness accrues to information-theoretic metrics like those discussed earlier in this paper.

5 Discussion and Future Work

It is well-known that geometry can be applied to information theory, and that the generalization performance of any single non-parametric supervised learning algorithm, even a non-parametric one, is governed by an inner product formula [11, 10]. In addition, whereas the NFL theorems and associated Bayesian machinery provide us with the mathematics governing the generalization performance of any single learning algorithm considered in isolation, little is currently known of the mathematics relating pairs of learning algorithms (cf. discussion of "head-to-head minimax distinctions" between algorithms in [10].) Accordingly, it is natural to investigate what geometric structure governs the relation between pairs of supervised learning algorithms.

In this paper it is shown that if one is provided with a loss function, that function naturally defines a metric over the space of supervised learning algorithms.

Intuitively, this metric measures the fraction of targets for which the difference in the expected performance of the two algorithms differs significantly.

Bounds on the value of this metric are then calculated for the case of binary outputs and 0-1 loss. These bounds establish that any two algorithms are almost exactly identical. In particular, even for small input spaces and training sets, for less than $2e^{-50}$ of all targets will the difference in expected generalization performance of *any* two algorithms exceed 1%. Analogously, for a uniform prior over targets, for fewer than $2e^{-50}$ of all training sets will the difference in posterior generalization error exceed 1%.

Intuitively, these results reflect the fact that the set of targets that appear "random" with respect to the two algorithms always swamps the set of targets that appear to have some structure. This is almost intuitively obvious when (for example) the two learning algorithms are the rules (always guess the input-output function h_1 = all 1's) and (always guess h_2 = all 0's) — the vast majority of targets have about as many 0's as 1's. What's less obvious is that there is always such a "random ... swamping" set of targets for other not so divergent h_i, and even for learning algorithms whose guessed hypotheses vary — perhaps wildly — with varying training sets.

The bounds calculated in this paper can also be viewed as providing us with (usually severe) restrictions on the probability distribution over targets, and therefore as providing us with (usually a large amount of) information concerning targets. As an example the simple English phrase "I expect that algorithm A will perform with an accuracy of at least 75%" conveys 20,000 bytes of information concerning the target, in that it says that the target must support a difference in generalization performance between A and the algorithm that always guesses A's opposite of at least 50%.

Posit however that we are given a description of the target, like "the target is well-described by a shallow decision tree", which together with our prior knowledge leads us to *conclude* (rather than merely presume) that A will perform with an accuracy of at least 75%. For this to happen we will usually have information concerning the relation between f and many other algorithms besides A and its opposite. One would expect that all that information far exceeds the original 20,000 bytes.

In addition to investigating this hypothesis, there are a number of other ways the work in this paper can be extended. Some of these are mentioned above in the introduction. As an example of another kind of extension, consider the case of the two learning algorithms $A \equiv$ "always guess h_1" and $B \equiv$ "always guess h_2" for two fixed hypothesis input-output functions h_1 and h_2. Then the value of the target for that set Ω of those inputs x where h_1 and h_2 agree are irrelevant as far as generalization performance is concerned. Then there are two $f(X \setminus d_X \setminus \Omega$ that maximize the (L^p-based) metric distance between the algorithms: $f = h_1$, and $f = h_2$. Future work involves extending this kind of reasoning to situations where there are many h_i, and the two algorithms choose among the h_i based on who agrees most with the training set (except that when there are ties they use different tie-breaking schemes). It may be that in certain broad scenarios the difference in the performance of two such algorithms is maximized for f equal to one of the h_i, just as for the case of learning algorithms based on only two h_i. As an example of the implications of this, say we have such a scenario, and are training by picking whichever of a pre-fixed set of neural nets agrees most often with the training set. Then it would follow that tie-breaking —which neural net you pick out of the set of nets all agreeing equally often with the training set— will have the most effect on expected cost if the target is itself one of those candidate neural nets.

Given a loss function and associated metric, one can also investigate issues like how distant from one another a set of learning algorithms should be for best performance when a meta-technique is used either to choose among them (e.g., as in cross-validation) or combine them (e.g., via ensembles, or stacking). This can be done either for a metric based on the flat $P(f)$ implicitly considered here or for an informative $P(f)$. One can also cluster currently popular learning algorithms, and perhaps determine something of the "islands" in algorithm space where the learning algorithms human being use tend to lie.

Given instead a full bilinear field over a manifold (i.e., a differential geometry metric), one can bring all of the machinery of differential geometry to bear on the topic of learning algorithms. For example, one can consider geodesics through the space of algorithms, and investigate how much the modifications to learning algorithms implemented by various meta-learning techniques (e.g., the setting of a hyperparameter via cross-validation) veer from such geodesics. In other words, one can investigate how much curvature is associated with the paths through algorithm-space of such meta-learning.

Perhaps most importantly though, the results of this paper spotlight a glaring gap in current understanding of the mathematics of supervised learning. By NFL, no supervised learning algorithm can be justified without a priori assumptions concerning the prior $P(f)$. In particular, this is true of boosting [3], support vector machines [4], bagging [2], stacking [1], cross-validation, and similar currently popular techniques. Indeed, by the results of this paper, the widespread popularity of those techniques implicitly reflects an extremely large amount of information imputed to the prior. (For some of those techniques, like cross-validation and boosting, the information is even more nuanced, concerning not just the prior but its relationship to the learning algorithms we humans tend to construct.) Yet to date, essentially none of this putative prior knowledge has formally codified.

Phrased differently, we act as though we have far more prior knowledge concerning learning scenarios, of a far more sophisticated sort, than that going into the usual mundane choices for $P(f)$, like "I believe the target is likely to be smooth", "has relatively few leaves", "has small coding length" or some such. Rather than in such forms, our extra information is implicit, being contained in our presumed knowledge of what learning techniques (and meta-techniques) work in the real world.[5] Clearly, it is imperative that this prior knowledge be accessed and then exploited to get maximal performance of our learning algorithms.

As an example of what form such an exploitation could take, say we start with the prior assumption that cross-validation would work well at choosing among a particular (!) set of candidate algorithms $\{A_i\}$. More precisely, given some real-world training set at hand d and (unknown) underlying f that generated d, we might assume that the algorithm chosen by using cross-validation on d and $\{A_i\}$, A', generalizes from d better than does the worst-performing of the algorithms in $\{A_i\}$. Now by the NFL theorems and the results of this paper, such an assumption cannot hold in general; for it to be valid, $P(f)$ must be (severely) restricted in some way. To date, nobody has even elaborated the general form such a restriction would take. Yet given the near-universal agreement that cross-validation works very well in the real world (i.e., works well with our favorite learning algorithms, on the real world's actual $P(f)$), one could argue that we have greater faith in the implicit restrictions placed by cross-validation on $P(f)$ than in any of the currently popular explicit restrictions on $P(f)$, like favoring f's representable as small trees. (Indeed, one often uses cross-validation to determine whether a bias towards small trees is appropriate!) If we could make those restrictions on $P(f)$ of cross-validation's explicit, then using them should allow us to construct a new Bayes-optimal algorithm B that not only outperforms any of the original $\{A_i\}$, but by being Bayes-optimal, outperforms cross-validation as well. More ambitiously, one might even try to combine the restrictions on $P(f)$ associated with many different learning techniques (i.e., not restrict attention to cross-validation), and thereby produce an algorithm which outperforms any of those techniques.

[5]Despite Bayesian lore to the contrary, one need not use the prior $P(f)$ to extend knowledge of a likelihood $P(d|f)$ into a full-blown posterior; a prior is not required to do Bayesian inference. Certain kinds of knowledge concerning the performance of certain learning algorithms can be used instead. See [13].

Astoundingly though, whether in regard to cross-validation or any of the other currently popular non-Bayesian (meta)learning techniques, little to nothing is currently known about what $P(f)$ must be for that (meta)learning technique to perform well. (See [6, 9] for some preliminary work in this area.) Yet it is hard to imagine a more crucial issue in supervised learning.

Acknowledgements: I would like to thank Charlie Strauss for helpful comments.

References

[1] L. Breiman. Stacked regression. University of California, Dept. of statistics, TR 367, 1992.

[2] L. Breiman. Bagging predictors. Univesity of California, Dept. of Statistics, TR 421, 1994.

[3] L. Breiman. Bias, variance and arcing classifiers. University of California, Dept. of Statistics, Technical Report, 1996.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd ed.)*. Wiley and Sons, 2000.

[5] Choquet Bruhat et al. *Analysis, Manifolds and Physics.* North-Holland, 1982.

[6] R. Vilalta et al. *ICML 2000 workshop on What Works Well Where.* 2000.

[7] Jakob Hansen. Combining predictors. PhD thesis, Dept. of Computer Science, University of Aarhus, 2000.

[8] Robert Wald. *General Relativity.* University of Chicago Press, 1984.

[9] D. H. Wolpert. Bayesian back-propagation over i-o functions rather than weights. In *Advances in Neural Information Processing Systems VI.* Morgan Kauffman,, 1994.

[10] D. H. Wolpert. The lack of a prior distinctions between learning algorithms and the existence of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390,1391–1421, 1996.

[11] D. H. Wolpert, editor. *The Mathematics of Generalization.* Addison-Wesley, New York, 1996.

[12] D. H. Wolpert. On bias plus variance. *Neural Computation*, 9:1211–1244, 1996.

[13] D. H. Wolpert. Reconciling bayesian and non-bayesian analysis. In *Maximum Entropy and Bayesian Methods 1993*. Kluwer Academic Press, 1996.